1. A method for a computer system for providing communication between a first system and a second system, wherein the first system uses a first version of a protocol and the second system uses a second version of the protocol, the method comprising:

providing a first application on the first system;

making a function call to a socket Application Programming Interface (API) for the first version;

translating the function call to a translated function call wherein the translated function call uses raw sockets;

making a function call to the socket API for the translated function call that uses raw sockets; and

passing a packet to a stack for the first version of the protocol.

2. The method of claim 1, wherein the first version of the protocol is IPv4 (Internet Protocol version 4) and wherein the second version of the protocol is IPv6 (Internet Protocol version 6).

3. The method of claim 2, further comprising supplying IP headers.

4. The method of claim 2, further comprising supplying IP headers only once.

5. The method of claim 2, wherein the method is implemented by inserting an API level translator layer between a socket API layer and a TCP/IPv4 layer.

6. The method of claim 2, wherein the method is implemented without using an IPv6 stack.

7. The method of claim 4, further comprising passing the packet to a network card driver.

8. The method of claim 2, further comprising providing an alternate implementation for a reception-related IPv4 socket function, wherein the alternate implementation comprises:

receiving an incoming packet on a raw socket;

checking a source host to determine the proper destination for the incoming packet;

checking a port number for the incoming packet;

stripping a transport and IP headers from the incoming packet; and

passing a payload to a destination application.

9. The method of claim 2, further comprising performing tunneling of IPv6 packets over IPv4 routing infrastructure.

10. The method of claim 2, further comprising using a Name Resolver service to perform name to address resolution related functions.

11. The method of claim 10, wherein the Name Resolver service is configured to run on a separate host that includes an IPv4 stack and an IPv6 stack.

12. The method of claim 11, wherein name to address resolution functions of the Name Resolver service use the IPv6 stack.

13. The method of claim 12, wherein the Name Resolver service is further configured to receive a query from the first system, use the address resolution functions to obtain a record and send the record to the first system.

14. The method of claim 2, further comprising providing an alternate implementation for a sending-related IPv4 socket function, wherein the alternate implementation comprises:

intercepting an IPv4 socket API call to send a packet;

translating the IPv4 socket API call to use a raw socket;

providing transport and IPv6 headers;

calling a corresponding IPv4 socket API function for the raw socket; and
passing the packet to the stack.

15. The method of claim 14, further comprising performing tunneling of IPv6 packets over IPv4 routing infrastructure.

16. The method of claim 14, further comprising fragmenting the packet.

17. The method of claim 14, further comprising passing the packet to a network card driver.

18. A set of executable instructions for implementing a method in an IPv4 (Internet Protocol version 4) computer system for providing communication between the IPv4 system and an IPv6 (Internet Protocol version 6) system, the method comprising:

providing an IPv4 application on the IPv4 system;

making a function call to an IPv4 socket Application Programming Interface (API);

translating the function call to a translated function call wherein the translated function call uses raw sockets;

making another function call to the IPv4 socket API for the translated function call that uses raw sockets; and

passing a packet to an IPv4 stack.

19. The set of executable instructions of claim 18, wherein the method further comprises supplying IP headers only once.

20. The set of executable instructions of claim 18, wherein the method is implemented by inserting an API level translator layer between a socket API layer and a TCP/IPv4 layer.

21. The set of executable instructions of claim 18, wherein the method is implemented without using an IPv6 stack.

22. The set of executable instructions of claim 18, wherein the method further comprises providing an alternate implementation for a reception-related IPv4 socket function, wherein the alternate implementation comprises:

receiving an incoming packet on a raw socket;

checking a source host to determine the proper destination for the incoming packet;

checking a port number for the incoming packet;

stripping a transport and IP headers from the incoming packet; and

passing a payload to the IPv4 application.

23. The set of executable instructions of claim 22, further comprising a computer-readable medium for storing the executable instructions.

24. The set of executable instructions of claim 18, wherein the method further comprises providing an alternate implementation for a sending-related IPv4 socket function, wherein the alternate implementation comprises:

intercepting an IPv4 socket API call to send data;
translating the IPv4 socket API call to use a raw socket;
providing transport and IPv6 headers;
calling a corresponding IPv4 socket API function for the raw socket; and
passing the data to the IPv4 stack.

25. The set of executable instructions of claim 24, wherein the method further comprises performing tunneling of IPv6 packets over IPv4 routing infrastructure.

26. The set of executable instructions of claim 24, wherein the method further comprises fragmenting the packet.

27. The set of executable instructions of claim 24, wherein the method further comprises passing the packet to a network card driver.

28. A system for enabling an IPv4 (Internet Protocol version 4) application to communicate across a computer network using an IPv6 (Internet Protocol version 6) system, the system comprising:

a computing device;

executable instructions executable on the computing device, wherein the executable instructions are configured to implement a method comprising:

making a function call to an IPv4 socket Application Programming Interface (API);

translating the function call to a translated function call wherein the translated function call uses raw sockets;

making another function call to the IPv4 socket API for the translated function call that uses raw sockets; and

passing a packet to an IPv4 stack.

29. The system of claim 28, further comprising an API level translator layer between a socket API layer and a TCP/IPv4 layer.

30. The system of claim 28, wherein the method further comprises providing an alternate implementation for a reception-related IPv4 socket function, wherein the alternate implementation comprises:

receiving an incoming packet on a raw socket;

checking a source host to determine the proper destination for the incoming packet;

checking a port number for the incoming packet;

stripping a transport and IP headers from the incoming packet; and

passing a payload to the IPv4 application.

31. The system of claim 28, wherein the method further comprises providing an alternate implementation for a sending-related IPv4 socket function, wherein the alternate implementation comprises:

intercepting an IPv4 socket API call to send the packet;
translating the IPv4 socket API call to use a raw socket;
providing transport and IPv6 headers;
calling a corresponding IPv4 socket API function for the raw socket; and
passing the packet to the IPv4 stack.

32. The system of claim 28, wherein the method further comprises performing tunneling of IPv6 packets over IPv4 routing infrastructure.

33. The system of claim 28, wherein the method further comprises fragmenting the packet.